Enterprise PL/I for z/OS and OS/390

Compiler and Run-Time Migration Guide

Version 3 Release 1

GC27-1458-00

Enterprise PL/I for z/OS and OS/390

Compiler and Run-Time Migration Guide

Version 3 Release 1

GC27-1458-00

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix A, “Notices” on page 35.

Third Edition (November 2001)

This edition applies to Version 3 Release 1 of Enterprise PL/I for z/OS and OS/390, 5655-H31, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department BWE/H3
P.O. Box 49023
San Jose, CA, 95161-9023
United States of America

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Introduction

This book contains information to help you migrate applications from previous releases of PL/I to Enterprise PL/I and OS/390 Language Environment. It suggests solutions to problems that arise because of differences in support between previous releases of PL/I (OS PL/I and PL/I for MVS & VM) and Enterprise PL/I.

IMPORTANT

The information in this book discusses migration considerations using Enterprise PL/I V3R1M0 and OS/390 V2R10 Language Environment or later. These two products must be installed in order to take advantage of the migration enhancements discussed in this book. The use of Enterprise PL/I will always refer to Version 3 Release 1.0 unless indicated otherwise. The use of Language Environment will always refer to OS/390 V2R10 Language Environment or later unless indicated otherwise.

This book is for system programmers, application programmers, and IBM support personnel who are involved in PL/I product migration. Prerequisite knowledge for using this book is:

- A general understanding of your operating system
- Some knowledge of the PL/I language and options
- Some knowledge of how PL/I uses Language Environment for its run-time environment

General concerns

This list of concerns is merely a representative list that reflects what has been important to some customers. It may not indicate what is important to any one individual customer. More details are provided in the rest of this book.

- Enterprise PL/I runs only under Language Environment 2.10 or later. If you have not migrated to Language Environment, then the migration task is more complicated.
- Enterprise PL/I has no support for VM.
- Enterprise PL/I has no support for multitasking (but it does support multithreading).
- Use of PDSE's is required unless either the prelinker is used or unless the compiler options NORENT LIMITS(EXTNAME(*n*)) (with *n* <= 8) are used.
- Code that is incorrect or invalid (for instance, code that uses uninitialized variables) may not run the same. This may not seem like an important problem, but it has been a significant issue for most of the customers that have migrated.
- Programs may need to be tuned for optimal performance. In particular, the use of the runtime option RPTSTG(ON), while useful when tuning, is much more costly now to leave on in a production program.
- Recompiling all your PL/I source is recommended; if this isn't done, the following options should be used:

  - CMPAT(V2) (or CMPAT(V1) is that's what currently being used with old PL/I)
  - DEFAULT(LINKAGE(SYSTEM))
  - LIMITS(EXTNAME(7))
  - NORENT
- Even if the options listed immediately above are used, there are some restrictions on mixing old and new object code:
  - CONTROLLED variables cannot be shared between old and new code.
  - FILE variables and constants cannot be shared between old and new code. However, a file written out by old code can be read by new - and vice versa.
  - The new code must be compiled with the NORENT option.
  - Whenever old code is used, all fetch/release restrictions from the older product apply. In particular, if a new MAIN does successfully FETCH and CALL an old module, then the old module cannot perform a subsequent FETCH of another module.
  - Old code, even if compiled with PL/I for MVS & VM, cannot FETCH a new module linked as a DLL.
  - For old code compiled with OS PL/I V2R3 or earlier:
    - An old MAIN not linked with LE cannot FETCH a new module.
    - A new MAIN cannot CALL or FETCH an old module unless either the old or new module is linked with SCEELKED and with INCLUDE SYSLIB(CEESG010).

Run-time environment for Enterprise PL/I

Enterprise PL/I uses Language Environment as its run-time environment. It conforms to Language Environment architecture and shares the run-time environment with other conforming languages such as C/370, C/C++, COBOL, and Fortran.

Language Environment is the common run-time environment for the following language compilers:

C/370
C/C++
COBOL for MVS & VM
COBOL for OS/390 & VM
Fortran
PL/I for MVS & VM
Enterprise PL/I

It provides a common set of run-time options and callable services. It also improves interlanguage communication (ILC) between high-level languages (HLL) and assembler by eliminating language-specific initialization and termination on each ILC invocation. Language Environment provides compatibility support for existing applications with a few restrictions.

Using your documentation

The publications provided with Enterprise PL/I are designed to help you program with PL/I. The publications provided with Language Environment are designed to help you manage your run-time environment for applications generated with Enterprise PL/I. Each publication helps you perform a different task.

The following tables show you how to use the publications you receive with Enterprise PL/I and Language Environment. You'll want to know information about both your compiler and run-time environment. For the complete titles and order numbers of these and other related publications, see “Bibliography” on page 38.

PL/I information

Table 1. How to use Enterprise PL/I publications

To...	Use...
Evaluate Enterprise PL/I	Fact Sheet
Understand warranty information	Licensed Programming Specifications
Plan for and install Enterprise PL/I	Enterprise PL/I Program Directory
Understand compiler and run-time changes and adapt programs to Enterprise PL/I and Language Environment	Compiler and Run-Time Migration Guide
Prepare and test your programs and get details on compiler options	Programming Guide
Get details on PL/I syntax and specifications of language elements	Language Reference
Diagnose compiler problems and report them to IBM	Diagnosis Guide
Get details on compile-time messages	Compile-Time Messages and Codes

Language Environment information

Table 2. How to use OS/390 Language Environment publications

To...	Use...
Evaluate Language Environment	Concepts Guide
Plan for Language Environment	Concepts Guide Run-Time Migration Guide
Install Language Environment on OS/390	OS/390 Program Directory
Customize Language Environment on OS/390	Customization
Understand Language Environment program models and concepts	Concepts Guide Programming Guide
Find syntax for Language Environment run-time options and callable services	Programming Reference
Develop applications that run with Language Environment	Programming Guide and your language Programming Guide
Debug applications that run with Language Environment, get details on run-time messages, diagnose problems with Language Environment	Debugging Guide and Run-Time Messages
Develop interlanguage communication (ILC) applications	Writing Interlanguage Applications
Migrate applications to Language Environment	Run-Time Migration Guide and the migration guide for each Language Environment-enabled language

Chapter 2. Installation considerations

This chapter contains product information you need to know at installation time. It discusses differences in user exits and effects of Language Environment Abnormal Termination Exit.

The Language Environment run-time options that you might want to consider at installation time are described in “Differences in run-time options” on page 22.

Product information

Enterprise PL/I has renamed its parts so that, if you want to, you can install it in the same SMP/E zone as OS PL/I. To help you identify the elements of each product, the following table lists the name differences:

Table 3. PL/I element names

OS PL/I	PL/I for MVS & VM	Enterprise PL/I
IEL0AA	IEL1AA	IBMZPLI
IKJEN00n	IEL1IKJn	
IEL0nn	IEL1nn	IBMZnn
PLInnnnn	IEL1Mnnn	IBMZMnnn
PLIXnnn	IEL1nnn	IBMZnnn
PLIHELP	IEL1PLIH	IBMZPLIH

Language Environment must be available before you can compile, prelink, link-edit, and run a Enterprise PL/I application. If you attempt to compile a program before installing Language Environment, the program will not compile and a message will be generated. A STEPLIB concatenation for SCEERUN must be added in the compile. The details of the data sets and modules shipped with Enterprise PL/I and Language Environment can be found in one of the documents listed below. If you want to know the names of the data sets and modules, storage requirements, or other details specifically for installation planning, refer to one of these documents:

Enterprise PL/I Program Directory
OS/390 Program Directory
OS/390 V2R10 Language Environment Customization

There are additional requirements you need to be aware of before you begin to use Enterprise PL/I and Language Environment.

You must have access to Language Environment when you compile your Enterprise PL/I application. When you compile your application and you use existing JCL, be sure your STEPLIB or JOBLIB statement includes SCEERUN (Language Environment run-time library). You can use the IBMZC cataloged procedure to compile PL/I applications.

Your compile step should include the following:

```
//PLI      EXEC PGM=IBMZPLI,REGION=4000K
//STEPLIB  DD  DSN=&LNGPRFX..SIBMZCMP,DISP=SHR
//         DD  DSN=&LIBPRFX..SCEERUN,DISP=SHR
```

Reading about the cataloged procedures provided with Enterprise PL/I can help you understand the use of SCEERUN during compilation. “Using PL/I Cataloged Procedures” is a chapter in *Enterprise PL/I for z/OS and OS/390 Programming Guide*.

When you link-edit your Enterprise PL/I application with Language Environment and you use existing JCL, be sure your SYSLIB statement includes SCEELKED (Language Environment link-time library).

Language Environment also provides the SIBMCAL2 library which you must specify, in the SYSLIB dataset concatenation, before SCEELKED if you use PLICALLA or PLICALLB.

You must specify SYSLIB if you plan to use it. Do not include SYSLIB unless you are using a LINKLIB which already includes SCEELKED.

Considerations for using assembler user exits

The only Assembler user exit supported by Enterprise PL/I is the Language Environment user exit CEEBXITA. IBMBXITA and IBMFXITA are not supported. For a detailed parameter description for CEEBXITA, see *OS/390 Language Environment Programming Guide.*.

Specific considerations

- The PL1DUMP, PLIDUMP or CEEDUMP file for the dump output is treated as a process resource and must not be cleared during enclave termination.
- The OS PL/I abend exit IBMBEER is ignored under Language Environment. See “Differences in Condition Handling” on page 19 for information on how to force an abend under Language Environment.

For more information on assembler language user exits, see *OS/390 Language Environment Programming Guide.*.

Considerations for using high-level language user exits

The OS PL/I Version 2 High-Level Language (HLL) user exit IBMBINT is not supported. Enterprise PL/I MAIN load module supports only CEEBINT. The load module always contains a copy of CEEBINT, either the application-specific one or the default one provided by Language Environment.

If you write a CEEBINT exit in PL/I, it must be compiled with Enterprise PL/I. If the CEEBINT exit calls any PL/I routines, those routines must also be compiled with Enterprise PL/I.

Do not use the OPTIONS(MAIN) statement in the user exit.

Execution of the STOP statement in the CEEBINT exit will terminate the application.

Chapter 3. Compile-time considerations

This chapter describes compile-time considerations when your run-time environment is Language Environment. You'll find out what OS PL/I Version 1 source code is supported in Enterprise PL/I.

The major factors to consider before migrating to Enterprise PL/I are:

- There is no VM support.
- Multitasking is not supported; however, multithreading is supported.
- There is no support for compilations under TSO.

Consider recompiling your OS PL/I and PL/I for MVS & VM applications with Enterprise PL/I. Support for combining object and load modules from previous PL/I releases with Enterprise PL/I object is limited.

Mixing Object Levels

Support for mixing objects produced by Enterprise PL/I and previous versions in a single application is limited. If you attempt to mix old and new objects, it is strongly recommended that you use the following compiler options:

- CMPAT(V2) (or CMPAT(V1) if your old code was compiled with that)
- DEFAULT(LINKAGE(SYSTEM))
- LIMITS(EXTNAME(7))
- NORENT

The CMPAT(V2) (or CMPAT(V1)) option is needed where a string, array or structure is passed to or received from non-PL/I code. This occurs, for example, when a DB2 stored procedure written in PL/I is passed a string.

The following are not supported:

- FILE variables and constants cannot be shared between old and new code. However, a file written out by old code and can be read by new - and vice versa.
- Controlled variables cannot be shared between old and new.
- Entry variables cannot be shared unless the NORENT compiler option is used

Dependency on Language Environment

Language Environment must be available whenever you compile a PL/I application. Language Environment is the run-time environment for the Enterprise PL/I compiler.

Compile-time options not supported by Enterprise PL/I

Enterprise PL/I does not support the following compile-time options:

CONTROL	NOINCLUDE
DECK	SEQUENCE
ESD	SIZE
FLOW	SMESSAGE
LMESSAGE	

The following compile-time options have had the indicated suboptions dropped:

LANGLVL	NOSPROG/SPROG (SPROG is always in effect)
LIST	m,n
SYSTEM	CMS, CMSTPL

The meaning of the following compile-time options has changed:

INCLUDE	The old meaning of INCLUDE (enabling %INCLUDE statements without use of the macro preprocessor) is always in effect in Enterprise PL/I. The new INCLUDE option is used under OS/390 UNIX System Services to help the compiler find the right include file.
OFFSET	This option no longer produces an offset table in the listing, instead, this option determines how offsets are presented in the listing: NOOFFSET produces offsets from the compile unit, and OFFSET produces offsets from the procedure.

Compatibility considerations and restrictions

There are some behavioral differences between the Enterprise PL/I compiler and previous PL/I compilers. As a result, the Enterprise PL/I compiler imposes some restrictions or may produce different results than when using source code created for OS PL/I or PL/I for MVS & VM. These differences are described in the following sections.

OS PL/I Version 1 source code

Enterprise PL/I compatibility with OS PL/I Version 1 source code is supported with the following exceptions:

- CHARSET(48) and CHARSET(BCD) are no longer supported. Support for these options were dropped by OS PL/I Version 2; however, there is an IBM-supplied tool that will convert the source.
- Graphic DBCS varies slightly from old EGCS in that the shift-in and shift-out code points are fixed.
- Suffixes that follow string constants are not replaced by the preprocessor—whether or not these are legal PL/I suffixes—unless you insert a delimiter between the ending quotation mark of the string and the first letter of the suffix. For example:

```
%DCL (GX, XX) CHAR;
%GX='||FX';
%XX='||ZZ';
DATA = 'STRING'GX;
DATA = 'STRING'XX;
DATA = 'STRING' GX;
DATA = 'STRING' XX;
```

under Version 1 produces the source:

```
DATA = 'STRING'||FX;
DATA = 'STRING'||ZZ;
DATA = 'STRING' ||FX;
DATA = 'STRING' ||ZZ;
```

whereas, under Enterprise PL/I it produces:

```
DATA = 'STRING'GX;
DATA = 'STRING'XX;
DATA = 'STRING' ||FX;
DATA = 'STRING' ||ZZ;
```

- CHECK statement, CHECK prefix, and CHECK condition support was dropped by PL/I for MVS & VM.

ENTRY statement

The ENTRY statement is supported with the following restrictions:

- All parameters must be BYADDR. The default compiler options will force this action.
- RETURNS must be BYADDR if an aggregate. The default compiler options will force this action.
- Return value, if any, will be converted to the attributes in the most recent PROC or ENTRY statement with a RETURNS option.

The following example shows how returns value might be converted:

```
a: proc;                       /* return value is undefined           */
  b: entry returns(fixed bin); /* return value converted to fixed bin */
  c: entry;                    /* return value converted to fixed bin */
  d: entry returns(fixed dec); /* return value converted to fixed dec */
end;
```

Array expressions

An array expression is not allowed as an argument to user functions unless it's an array of scalars of known length.

The following example shows numeric array expressions supported in calls:

```
dcl x entry, (y(10),z(10)) fixed bin(31);

call x(y + z);
```

The following unprototyped call would be flagged since it requires a string expression of unknown size:

```
dcl a1 entry;
dcl (b(10),c(10)) char(20) var;

call a1(b || c);
```

However, the following prototyped call would not be flagged:

```
dcl a2 entry(char(30) var);
dcl (b(10),c(10)) char(20) var;

call a2(b || c);
```

Structure expressions

Structure expressions are supported in assignment statements, including BY NAME assignments, with the following exceptions:

- Structure expressions as arguments are not supported unless both of the following conditions are true:
  - There is a parameter description.
  - The parameter description specifies all constant extents.

DEFINED variables

Support for iSUB defining is limited to arrays of scalars.

Simple DEFINED variables are supported only for the following:

- Scalars
- Structures with constant extents matching those in the base variable
- Arrays of such scalars and structures as long a the array is not based on a controlled variable

When simple defining does not apply, string-overlay defining is assumed and flagged with an E-level message (as is true with PLIOPT).

DBCS

DBCS can be used only in the following:

- G and M constants
- Identifiers
- Comments

G literals can start and end with a DBCS quote followed by either a DBCS G or an SBCS G.

Stream I/O

Stream I/O is supported with the following restrictions:

- For PUT/GET DATA, the following restrictions apply:
  - DEFINED is not supported if the DEFINED variable is BIT or GRAPHIC or has a POSITION attribute.
  - DEFINED is not supported if its base variable is an array slice or an array with a different number of dimensions than the defined variable.

Record I/O

Record I/O is supported with the following exceptions:

- EVENT clauses on READ/WRITE are not supported.
- The UNLOCK statement is not supported.
- The BACKWARDS file attribute is not supported.
- Alternate index paths are not supported.
- Regional(2) and Regional(3) files are not supported.
- TRANSIENT files are not supported
- The EXCLUSIVE file attribute is not supported.
- The following options of the ENVIRONMENT attribute are not supported:
  ADDBUFF ASCII BUFFERS(n) BUFND BUFNI BUFOFF INDEXAREA(n) LEAVE
  NCP(n) NOWRITE REREAD REUSE SIS SKIP TOTAL TP({MIR}) TRKOFL

Built-in functions

Built-in functions are supported with the following exceptions/restrictions:

- The PLITEST built-in function is not supported.
- Pseudovariables are not supported in:
  - The STRING options of PUT statements
- Pseudovariables in DO loops are restricted to:
  - IMAG
  - REAL
  - SUBSTR
  - UNSPEC
- The POLY built-in function has the following restrictions:
  - The first argument must be REAL FLOAT.
  - The second argument must be scalar.
- The COMPLEX pseudovariable is not supported.

Batch compilations

Compilation is not performed in PROCESS-delimited chunks, and this difference has the following consequences:

- Options on later sets of PROCESS statements are ignored
- One TEXT deck or .o is produced
- One listing file with one set of messages is produced
- External variables with the same name must match

The following example demonstrates a batch compilation. In this case, the mismatches in ***b*** and ***x*** would be flagged.

```
*process opt(0);

 a: proc;
   dcl b ext entry(1,2 char(2), 2 char(2));
   dcl
     1 x ext,
       2 x1a char(2),
       2 x1b char(2);

   call b(x);
 end;

*process opt(2);

 b: proc(p);
   dcl p pointer;
   dcl
     1 x ext,
       2 x1a bit(16),
       2 x1b bit(16);

 end;
```

Miscellaneous unsupported elements

The following miscellaneous elements are not supported:

- NOMAP, NOMAPIN, and NOMAPOUT are accepted but ignored, even if parmlist/arguments are given.

Storage report changes

The PLIXHD variable is no longer used as the heading in storage reports. The identifier PLIXHD is no longer reserved; you can declare it and use it as you would declare and use any other variable.

Compiler messages

The messages issued by the Enterprise PL/I compiler are completely different from messages issued by previous PL/I compilers. For detailed descriptions of messages produced by Enterprise PL/I, see *Enterprise PL/I Compile-Time Messages and Codes*.

Messages that PL/I issues for errors in the PLIXOPT string

The PLIXOPT variable is a varying-length character string that contains run-time options you can specify at compile time. The messages that the compiler produces to diagnose errors in these options have changed. In most cases, the PL/I messages now list an associated Language Environment message that you should read for more information about the error.

PL/I parses the PLIXOPT string and produces the Language Environment CEEUOPT CSECT. If you explicitly include CEEUOPT in your recompiled application ahead of the compiler-generated CEEUOPT CSECT, the explicitly

included CEEUOPT CSECT overrides the one generated by the compiler for the options specified in the PLIXOPT string.

Chapter 4. Link-edit considerations

This chapter describes factors you must consider when you link-edit an object module produced by Enterprise PL/I.

Important: If you compile with the option RENT or with the option LIMITS(EXTNAME(*n*)) with *n* > 8, then you must either use the prelinker or use the binder and place your binder output in a PDSE. For more information about linking, see the chapter titled 'Link-editing and Running' in the *Enterprise PL/I Programming Guide*.

Using FETCH in your routines

You can FETCH Enterprise PL/I routines, OS/390 C DLLs, and Assembler routines except for the following restrictions:

- OPTIONS(FETCHABLE) must be specified on the PROCEDURE statement for the entry point of the fetched routine.
- Unless the NORENT option has been specified, the ENTRY declaration in the routine that FETCHes must not specify OPTIONS(COBOL) or OPTIONS(ASM)—these should be specified only for COBOL or ASM routines not linked as DLLs.
- OPTIONS(FETCHABLE) must be specified on the PROCEDURE statement for the entry point of the FETCHABLE routine or the procedure must be compiled with the DLLINIT option.
- Unless the NORENT option has been specified, a PROCEDURE specifiying OPTIONS(FETCHABLE) must be linked as a DLL.
- RMODE(24) routines cannot be FETCHed.

For a detailed description of these restrictions, see the chapter titled 'Link-editing and Running' in the *Enterprise PL/IProgramming Guide*.

Using PLICALLA or PLICALLB entry

For Enterprise PL/I programs that use PLICALLA or PLICALLB as the main entry point, link-edit the object modules with the SIBMCAL2 dataset in front of the SCEELKED dataset. See “PLICALLA considerations” on page 15 and “PLICALLB considerations” on page 16 for details.

ENTRY CEESTART requirement

If a Enterprise PL/I or PL/I for MVS & VM main procedure is link-edited with object modules produced by other language compilers or by assembler, and is the first module to receive control, the user must ensure that the entry point of the resulting executable program is resolved to the external symbol CEESTART. This happens automatically if the Enterprise PL/I or PL/I for MVS & VM main procedure is first in the input to the linkage editor. Run-time errors occur if the executable program entry point is forced to some other symbol by use of the linkage editor ENTRY control statement.

Restrictions on using the binder

You can use the binder in place of the prelinker and linkage-editor, with the following exceptions:

- CICS Prior to CICS 1.3, PDSEs are not supported. From CICS Transaction Server 1.3 onwards, there is support in CICS for PDSEs. Please refer to the CICS Transaction Server for OS/390 Release Guide, GC34-5701, where there are several references to PDSEs, and a list of prerequisite APAR fixes.
- MTF MTF does not support PDSEs. If your program targets MTF, you cannot use the binder.

Chapter 5. Run-Time Considerations

Before you migrate to Language Environment or Enterprise PL/I, you should read this chapter. It discusses the functional differences between previous PL/I compilers and Enterprise PL/I and its run-time environment Language Environment. These differences should be considered before you install Language Environment or Enterprise PL/I. Other chapters in this book discuss differences you must consider during and after installation.

Factors to consider before migrating to Enterprise PL/I are:

- There is no support for multitasking.
- There is no support for IBMBSIR or IBMBHKS.

Differences in PLICALLA and PLICALLB Support

The interfaces in the following sections are not recommended for use in Enterprise PL/I. They are supported only for compatibility reasons.

PLICALLA considerations

Language Environment provides support for Enterprise PL/I applications that use the PLICALLA entry point. It also provides support for recompiled OS PL/I and PL/I for MVS & VM applications that want to continue to use PLICALLA as the primary entry point.

When you recompile your OS PL/I or PL/I for MVS & VM program with Enterprise PL/I, there is no need to INCLUDE Language Environment-provided PLISTART CSECT when you link your main load module. You just need to make sure the SIBMCAL2 dataset is concatenated in front of the SCEELKED dataset. If you don't do this, the linkage editor or loader issues an error message for an unresolved ENTRY PLICALLA statement.

You can also use PLICALLA as the primary entry point of a FETCHed/CALLed main load module; however, the calling routine must pass only user arguments which are passed to a subroutine. If run-time options are passed, they are treated as user arguments.

If you develop a new application in Enterprise PL/I and you want the main procedure to receive user arguments like a subroutine, do one of the following:

- Receive control directly from IMS by:
  - Using CEESTART as the primary entry point of the load module.
  - Specifying the SYSTEM(IMS) compile-time option.
- Receive control from an assembler program or a procedure using a FETCH or CALL statement by:
  - Using CEESTART as the primary entry point of the load module.
  - Specifying the NOEXECOPS option and the SYSTEM(MVS) compile-time option.

Language Environment support of PLICALLA is not available in the following environments:

CICS environment
Preinitialized environment
Nested enclave environment.

Passing parameters

PLICALLA can be used under IMS, but recall that if a MAIN procedure is recompiled with Enterprise PL/I using SYSTEM(IMS), all parameters must be POINTERs.

Table 4 provides the expected argument passing convention (either BYADDR or BYVALUE) when the main procedure of your OS PL/I PLICALLA application is recompiled with Enterprise PL/I. Note that SYSTEM(CICS) is not listed in this table since PLICALLA cannot be used under CICS with Enterprise PL/I

Table 4. Parameter passing for the main procedure compiled with Enterprise PL/I

System environment	Invoked from IMS[1]	Invoked from assembler program[2]	Invoked by PL/I FETCH/CALL Statement[2]
SYSTEM(MVS)	BYADDR	BYADDR	BYADDR
SYSTEM(IMS)	BYVALUE[3]	Not supported	Not supported
SYSTEM(TSO)	BYADDR	BYADDR	BYADDR

[1]LANG=PL/I must be specified and it passes indirect by reference.
[2]It must have already passed indirect by reference or by value.
[3]PL/I library will convert the argument list to direct by value.

PLICALLB considerations

Language Environment provides support for recompiled OS PL/I or PL/I for MVS & VM PLICALLB applications that continue to use PLICALLB as the primary entry point. The following list shows the PLICALLB parameter mapping for Language Environment:

- Address of argument list (argument must either point to an address or be zero)
- Address of the length of ISA storage mapped to STACK(*init_size*)
- Address of ISA storage used as the initial STACK segment
- Address of the options word in which the run-time options for a program are specified. These options are: RPTSTG, TRAP, HEAP(,,KEEP|FREE)|(,,ANY|BELOW). The hexadecimal value for each option is defined as follows in the assembler program:

```
OPTIONS DC AL1(RPTSTG+TRAP,FREEHEAP+ANYHEAP,0,0)
*
RPTSTG     EQU X'80'
RPTSTGOFF  EQU X'40'
TRAP       EQU X'20'
TRAPOFF    EQU X'10'
*
KEEPHEAP   EQU X'20'
FREEHEAP   EQU X'10'
ANYHEAP    EQU X'08'
BELHEAP    EQU X'04'
```

- Address of HEAP storage length for a program is mapped to HEAP(*init_size*)
- Address of HEAP storage is used as the initial HEAP segment
- Address of HEAP increment for a program is mapped to HEAP(,*incr_size*)
- Address of ISA increment for a program is mapped to STACK(,*incr_size*) (optional) is mapped to NONIPTSTACK(,*incr_size*)

When the above argument list is passed in via the PLICALLB entry point, the argument in the list must either point to an address or be zero. The hight-order bit of an argument must be ON to indicate the end of the argument list. R1 must contain the address of the argument list.

With Language Environment, the run-time options passed via the PLICALLB entry point are processed as options specified on invocation of the application and have a higher precedence than CEEUOPT or PLIXOPT options. The assembler user exit cannot be used to alter the run-time options passed through the PLICALLB invocation.

To summarize, the run-time options passed in have the following precedence (from highest to lowest) among Language Environment option specification methods:

1. Options specified via the PLICALLB entry point
2. Options specified in the PLIXOPT string or in CEEUOPT

The user arguments passed to the PL/I main routine have the following precedence (from highest to lowest):

1. Output from CXIT_PARM or AUE_PARM of the assembler user exit
2. User arguments passed in via the PLICALLB entry

Note: The input to CXIT_PARM or AUE_PARM of the assembler user exit is the first argument in the PLICALLB parameter list; that is, the address of a vector of user argument addresses.

Language Environment encourages the use of above-16M-line storage. For compatibility with OS PL/I, Language Environment maps the user-supplied ISA and HEAP storage to STACK and HEAP. With this mapping, however, Language Environment still needs to issue some GETMAINs. Since user-supplied ISA/HEAP storage is usually below the 16M line, below-16M-line storage can be quickly consumed under Language Environment. How Language Environment manages storage is described in the *OS/390 Language Environment Programming Guide*.

Language Environment manages storage differently than OS PL/I. It divides storage into more categories than the OS PL/I-supported ISA and HEAP.

Language Environment allocates below-16M-line storage using the init_sz24 and incr_sz24 suboptions specified in the HEAP option.

When you develop new applications in Enterprise PL/I and want to pass both run-time options and arguments to a PL/I main procedure, especially to provide user-supplied stack and heap storage from an assembler program, take advantage of Language Environment's preinitialization support as described in *OS/390 Language Environment Programming Guide*.

Language Environment support of PLICALLB is not available in the following environments:

CICS
IMS
Preinitialized environment
Nested enclave environment

Passing parameters

OPTIONS(BYADDR) passes the argument indirectly by reference and is the usual argument-passing convention. Enterprise PL/I also provides OPTIONS(BYVALUE) which passes arguments directly by value.

You must use the BYADDR option when you want to pass parameters using PLICALLB. PLICALLB is invoked from assembler which passes the argument list indirectly by reference.

Table 5 provides the expected argument passing convention (either BYADDR or BYVALUE) when the main procedure of your OS PL/I PLICALLB application is recompiled with Enterprise PL/I:

Table 5. Parameter passing for the main procedure compiled with Enterprise PL/I

System environment	Invoked from assembler program[1]
SYSTEM(MVS)	BYADDR
SYSTEM(CMS|CMSTPL)	BYADDR
SYSTEM(CICS)	Not supported
SYSTEM(IMS)	Not supported
SYSTEM(TSO)	BYADDR

[1]It passed the argument list required by the PLICALLB entry.

Differences in preinitialization support

The PL/I preinitialized program interface is not supported for Enterprise PL/I applications. Use the Language Environment preinitialization service as described in the *OS/390 Language Environment Programming Reference*.

Differences in DATE/TIME built-in functions

The DATETIME and TIME built-in functions now return the number of milliseconds in all environments. The syntax and description of these built-in functions are in the Enterprise PL/I for OS/390 *PL/I Language Reference*.

Differences in user return codes

Enterprise PL/I and Language Environment support a FIXED BIN(31) four-byte user return code value for PLIRETC, PLIRETV, and OPTIONS(RETCODE). This support removes the old restriction that the maximum value had to be <= 999.

The following table shows how PL/I user return code is supported:

Table 6. Return code behavior under Language Environment

Function	OS PL/I load module	OS PL/I object module linked with Language Environment	Enterprise PL/I load module
PLIRETC built-in function	2-byte value with maximum <= 999	4-byte value with maximum < 2G	4-byte value with maximum < 2G
PLIRETV built-in function	2-byte value	Lower 2 bytes of a 4-byte value	4-byte value
RETCODE option	Lower 2 bytes of R15	Lower 2 bytes of R15	2-byte value

For PLIRETC, Enterprise PL/I and relinked OS PL/I load modules can set a 4-byte user return code value.

For PLIRETV and RETCODE, Enterprise PL/I load modules can receive a 4-byte user return code value.

Under Language Environment, the PL/I user return code is always reset to zero upon return from the PLISRTx invocation. This is not the case with OS PL/I run-time.

Differences in Condition Handling

PL/I condition handling semantics remain supported under Language Environment; however, the timing of issuing the run-time message for an ERROR condition with respect to the ERROR ON-unit is different in the following way:

- The run-time message for an ERROR condition is issued only if there is no ERROR ON-unit established, or if the ERROR ON-unit does not recover from the condition by using a GOTO out of the ERROR ON-unit. Thus you can use a GOTO out of the ERROR ON-unit to avoid a message for a PL/I ERROR condition.

For other PL/I conditions whose implicit action includes printing a message and raising the ERROR condition, the message is issued before control is given to an established ERROR ON-unit.

Table 7 shows when the run-time message for an ERROR condition is issued under OS PL/I with respect to the ERROR ON-unit.

Table 7. OS PL/I Version 2 Release 3 ERROR ON-unit and message for an ERROR condition

Condition	No ON-units	ERROR ON-unit No GOTO	ERROR ON-unit GOTO
ERROR condition raised[1]	Message	Message prior to ON-unit	Message prior to ON-unit
ZERODIVIDE condition raised[2]	Message	Message prior to ON-unit	Message prior to ON-unit

Notes:

1 Taking the square root of a negative number, data exception, etc.

2 With no ZERODIVIDE ON-unit; thus, implicit action is taken. Message is printed, ERROR condition is raised.

Table 8 shows when the run-time message for an ERROR condition is issued under Language Environment with respect to the ERROR ON-unit.

Table 8. Language Environment ERROR ON-unit and message for an ERROR condition

Condition	No ON-units	ERROR ON-unit No GOTO	ERROR ON-unit GOTO
ERROR condition raised[1]	Message	Message after ON-unit	No message
ZERODIVIDE condition raised[2]	Message	Message prior to ON-unit	Message prior to ON-unit

Notes:

1 Taking the square root of a negative number, data exception, etc.

2 With no ZERODIVIDE ON-unit; thus, implicit action is taken. Message is printed, ERROR condition is raised.

The SNAP traceback message produced by ON ERROR SNAP continues to be issued before the ERROR ON-unit receives control. The SNAP traceback message is not identical to the regular ERROR message.

Some program return code modifiers have changed under Language Environment, depending upon what compiler was used. The behaviors are:

- **OS PL/I V2R3**:

  A return code of 2000 is added for the case where the ERROR condition is raised and the program terminates without returning from an ERROR or FINISH ON-unit.

- **Enterprise PL/I and PL/I for MVS & VM**:

  A return code of 3000 is added for severity 3 conditions (severe error—abnormal termination).

Most PL/I conditions are severity 3, with the following severity 1 exceptions: ENDPAGE, FINISH, NAME, PENDING, STRINGRANGE, STRINGSIZE, UNDERFLOW, ATTENTION signaled, CONDITION signaled.

Note: Above information is useful when using the Language Environment ERRCOUNT run-time option.

If your OS PL/I application used to force an abend for an unhandled condition under OS PL/I run-time using OS PL/I assembler user exit IBMBXITA or abend exit IBMBEER, use the following ways to force an abend under Language Environment:

- Run your application with the Language Environment ABTERMENC(ABEND) option. You cannot specify your own abend code via the run-time option.
- Use Language Environment assembler user exit CEEBXITA to force an abend with your own abend code.

For ZERODIVIDE, OVERFLOW, and SIZE, the ERROR condition is raised if the condition goes unhandled.

The FOFL condition is not raised for FIXED BIN. It is raised only for FIXED DEC and decimal PICTURE.

Language Environment provides limited support for OS PL/I IBMBXITA and IBMBEER. See “Considerations for using assembler user exits” on page 5 for details.

An UNHANDLED condition of severity 2 or higher now produces an abend U4039 and optionally a system dump if SYSUDUMP or SYSABEND ddname is present. If ABTERMENC(RETCODE) is in effect, your application continues the termination with an abend code. If you don't want to see the U4039 abend, Language Environment provides you the facilities to suppress it.

See “Abnormal Termination Exit” in *OS/390 Language Environment Customization* for ways to suppress or change the U4039 abend.

Differences in run-time messages

The format and content of run-time messages are different. If you have applications that analyze run-time messages, you must change the applications to allow for the differences. The differences include:

- The message number in the message prefix is four digits instead of three digits in the form IBM*nnnnx*, where *nnnn* represents the message number and *x* represents the severity of the message.
- The message severity in the message prefix can be I, W, E, S, or C.
- The message text of some messages has been enhanced.

Details are provided in *OS/390 Language Environment Debugging Guide and Run-Time Messages*.

Under Language Environment, run-time messages go to the MSGFILE destination specified in the run-time option MSGFILE. The default for MSGFILE destination is SYSOUT. The user output still goes to SYSPRINT. MSGFILE(SYSPRINT) is not supported under Enterprise PL/I. For more information about the MSGFILE option, refer to *OS/390 Language Environment Programming Guide*.

Under Language Environment, run-time messages give offset values that are relative to the start of the external procedure, rather than relative to the start of the block that contains the statement. You can use these offsets to help you find the statement that is in error. To do this, match the offset provided in the message with the offset given in the pseudo-assembler listing that the compiler produces when you specify the LIST compile-time option.

Differences in PLIDUMP

PLIDUMP now produces a Language Environment-style dump. The way you use PLIDUMP and the dump output is different. The following list contains the differences in the way you use PLIDUMP and the output produced. *Compile unit* refers to the primary entry point of the external procedure and *Compile unit name* refers to the name of the external procedure.

- The ddname of the dump output file can be CEEDUMP, PLIDUMP, or PL1DUMP. If you do not define one of these files, Language Environment creates a default CEEDUMP file to contain the dump output. The LRECL of the dump output file must be at least 133 bytes to prevent dump records wrapping, not the 121 bytes required by OS PL/I. If you write the dump output

to the SYSOUT file, make sure you change the default LRECL size of 121 to 133 to prevent wrapping. Use LRECL of 137 for variable-length files.

- When you use the hexadecimal (H) option of PLIDUMP, you must specify the ddname CEESNAP; otherwise, no SNAP dump will be produced.

  When you specify the hexadecimal (H) option under OS/390, the output from SNAP includes all system control program information (SDATA=ALL). OS PL/I provides only partial information (SDATA=CB, Q, and TRT).
- The dump output contains information related to other languages (for example, C/C++ or COBOL).
- The identifying character string is limited to 60 bytes rather than the 90 bytes OS PL/I supported.
- The traceback section lists the compile-unit name associated with each entry point name. When the entry point is a secondary entry point, the primary entry point name associated with the actual entry point is not listed.

  The traceback section also contains offsets relative to the address of the *compile unit*, as well as offsets relative to the address of the real entry point.
- Run-time messages are in a separate section; they are no longer part of the traceback section.
- When you specify the BLOCK (B) option of PLIDUMP, the condition handler save areas appear in the block section of the dump. If you do not specify the BLOCK option of PLIDUMP, the condition handler save areas do not appear in the dump.
- If the program was compiled with the TEST compile-time option, the BEGIN blocks that are ON-units are identified as _ON_Begin_line_Blk_number while other BEGIN blocks are identified as _Begin_line_Blk_number where line is the line number where the Begin block begins and number is block for the begin-block.
- PL/I library routines are identified by name in the dump.
- Assembler routines that conform to the rules for mimicking PL/I routines are identified by their CSECT names in the dump output.
- PLIDUMP now conforms to National Language Support standards.
- PLIDUMP can supply information across multiple Language Environment enclaves. For example, if an application running in one enclave FETCHes a main procedure (an action that creates another enclave), PLIDUMP contains information about both procedures.

Differences in run-time options

Language Environment run-time options replace OS PL/I run-time options. Most OS PL/I run-time options have an equivalent Language Environment run-time option that provides the same function. This section describes differences in the use of run-time options.

Pre-Language Environment storage was initialized to zero. By default Language Environment does not do this and it can be a problem for programs with uninitialized variables. One way to handle this situation is to use the run-time option STORAGE by using the third parameter to initialize all storage to zero. Note

that the use of this method has serious performance costs, and modifying the program so that all variables are initialized is the preferred solution.

You should adapt your applications to allow for the following differences:

- The Language Environment ABTERMENC option controls which type of return/abend code your application receives at abnormal termination. ABTERMINC(RETCODE) allows your application to receive a run-time return code, which is equivalent to the way OS PL/I worked.
- The Language Environment ERRCOUNT option limits the number of errors that are handled at run-time. ERRCOUNT(0) specifies that there is no limit, which is equivalent to the way OS PL/I worked.
- The Language Environment DEPTHCONDLMT option limits the extent to which conditions can be nested. To maintain compatibility, specify DEPTHCONDLMT(0), which means there is an unlimited depth.
- The OS PL/I COUNT option is ignored.
- The OS PL/I FLOW option is ignored.
- The OS PL/I HEAP option is always in effect. This means that when you allocate storage for BASED and CONTROLLED variables, the storage always comes from HEAP storage. The storage does not come from a PL/I Initial Storage Area (ISA). HEAP(0) is not supported and, if used, is ignored.
- The Language Environment NATLANG option replaces the OS PL/I LANGUAGE option.
- The Language Environment RPTSTG option replaces the OS PL/I REPORT option.
- The Language Environment TRAP option replaces both OS PL/I SPIE and STAE options. The following table shows how the OS PL/I SPIE and STAE options map to Language Environment's TRAP option:

Table 9. Mapping of SPIE and STAE options to the TRAP option

OS PL/I	Language Environment	Action
SPIE | NOSPIE STAE | NOSTAE	TRAP(ON,SPIE) TRAP(OFF)	If either SPIE or STAE is specified or defaulted in input, TRAP is set to TRAP(ON,SPIE). If both NOSPIE and NOSTAE are specified, TRAP is set to TRAP(OFF). TRAP(ON,SPIE) is the recommended setting.

Note: Applications performing their own condition management often conflict with Language Environment condition management. See your *OS/390 Language Environment Programming Guide* for more information on Language Environment condition handling.

- The Language Environment STACK option replaces both OS PL/I ISASIZE and ISAINC options. You do not need to change source code that contains ISASIZE and ISAINC. In addition, object modules and/or load modules containing the PLIXOPT string will run under Language Environment with the ISASIZE and ISAINC honored as before.

  Use STACK(,,ANY) for your Enterprise PL/I application. Your application must run in AMODE(31) to use STACK(,,ANY).

  Under CICS, ALL31(ON) and STACK(,,ANY) are the defaults.

- The Language Environment Environment XUFLOW option determines if the UNDERFLOW condition is raised when underflow occurs. XUFLOW(AUTO) preserves PL/I semantics with regard to raising the UNDERFLOW condition.

For more information about run-time options, see the *OS/390 Language Environment Programming Reference*.

For OS PL/I applications, the options specified in the PLIXOPT string are processed as the application-specific options. Do not mix PLIXOPT and CEEUOPT.

Differences in storage report

The format, contents, and destination of the run-time storage report have changed. Language Environment provides storage information equivalent to OS PL/I. The details of storage report is described in *OS/390 Language Environment Programming Reference*.

The PLIXHD declaration is no longer used to provide the heading for the run-time storage report. Instead, use Language Environment's Callable Service, CEE3RPH, to specify the heading.

Differences in interlanguage communication support

Recompilation of PL/I modules in ILC applications containing OS PL/I or PL/I for MVS & VM with Enterprise PL/I is recommended.

ILC between Enterprise PL/I and the following languages is not supported:

- Fortran (prior to Language Environment Release 5)
- OS/VS COBOL
- VS COBOL II Version 1 Release 2 or earlier releases

For more information, see *Language Environment for OS/390 & VM Writing Interlanguage Communication Applications*.

OPTIONS(COBOL) is treated like OPTIONS(ASM). There is no remapping of parameters via MAPIN or MAPOUT. This is both a compile-time difference and a run-time difference, but it will become apparent only at run-time.

The behavior of certain applications that use ILC might be different. For example:

- Condition handling might behave differently. The major causes of differences in condition handling are that the INTER option is now ignored, and that PL/I condition handling facilities can deal with conditions occurring in non-PL/I routines whether or not you specify INTER.
- Under OS PL/I, in applications that used ILC, the environment initialization and termination of the involved languages, including PL/I, could occur multiple times. With Language Environment, there is only one run-time environment, and language-specific initialization and termination occurs only once. Changes in behavior that you might see include opening and closing of files, releasing of allocated storage, and invocation of established ON-units.

  Note: If you have designed your own code to manage your run-time environments, you should remove it as part of your migration efforts.

This *private* code is incompatible with Language Environment and will conflict with the run-time environment.

For a complete description of how ILC works in the Language Environment run-time environment, see *Language Environment for OS/390 & VM Writing Interlanguage Communication Applications*.

Differences in assembler support

With Enterprise PL/I, the object module contains the CSECT name CEESTART. It also contains CEEMAIN if it has OPTIONS(MAIN) or CEEFMAIN if it has OPTIONS(FETCHABLE). Enterprise PL/I no longer produces PLISTART and PLIMAIN CSECTs. CEESTART, CEEMAIN, and CEEFMAIN are not supported as a standard entry point and you cannot call them directly from an assembler program. You can call CEESTART from an assembler program only when it is a CSECT name of a Enterprise PL/I routine statically linked with an assembler program. Therefore, any assembler program mimicking a OS PL/I main procedure (calling PLISTART directly as a standard entry point), must continue to use PLISTART under Language Environment.

With Language Environment, assembler programs that call a PL/I routine must follow the calling conventions defined by Language Environment. For example, Register 13 pointing to a save area, save areas properly back-chained, and the first word of the save area being zero. For detailed information, see *OS/390 Language Environment Programming Guide*.

If your OS PL/I main program is called by an assembler program and you want to convert your assembler program to use Language Environment-conforming assembler, you must recompile your OS PL/I program with Enterprise PL/I without OPTIONS(MAIN). The called Enterprise PL/I program is treated as a subroutine and runs under the same Language Environment enclave where the assembler program is the main program and the called Enterprise PL/I program is a subroutine.

Your Language Environment-conforming assembler main program must explicitly include the Language Environment-Enterprise PL/I signature CSECT, CEESG011, when calling a PL/I subroutine to ensure the Language Environment-PL/I-specific run-time environment is initialized. There are three ways Language Environment-conforming assembler routines can pass control to a Enterprise PL/I subroutine:

1. Branch to a statically-linked Enterprise PL/I subroutine.
2. Use the Language Environment macro CEEFETCH to branch to a separately-linked Enterprise PL/I subroutine.
3. Use assembler instructions such as LOAD and BALR to branch to a separately-linked Enterprise PL/I subroutine.

When you use method 1 or 2 with Enterprise PL/I, you don't need to include CEESG011 with your assembler program. If your assembler program uses instructions as described in method 3, you must always include CEESG011 with your assembler program.

Condition handling of the LINK from assembler is now clearly defined. For detailed information, see *OS/390 Language Environment Programming Guide* and *Enterprise PL/I Programming Guide*.

Differences in language element behavior

There are also some language elements that can cause your program to run differently under Enterprise PL/I than it does under PL/I for MVS & VM due to differences in the hardware or in the implementation of the language by the compiler. Each of the following items is described in terms of its Enterprise PL/I behavior.

FIXED BIN(p) maps to one byte if p <= 7

If you have any variables declared as FIXED BIN with a precision of 7 or less, they occupy one byte of storage under Enterprise PL/I instead of two as under PL/I for MVS & VM and earlier. If the variable is part of a structure, this usually changes how the structure is mapped, and that could affect how your program runs. For example, if the structure were read in from a file, fewer bytes would be read in under Enterprise PL/I than under PL/I for MVS & VM or earlier PL/I release.

To avoid this difference, you could change the precision of the variable to a value between 8 and 15 (inclusive).

INITIAL attribute for AREAs is ignored

The Enterprise PL/I compiler ignores the INITIAL attribute for AREAs, and you should convert any INITIAL clauses for AREAs into assignment statements.

For example, in the following code fragment, the elements of the array are not initialized to a1, a2, a3, and a4:

```
dcl (a1,a2,a3,a4) area;
dcl a(4) area init( a1, a2, a3, a4 );
```

However, you can rewrite the code as follows so that the array is initialized as desired:

```
dcl (a1,a2,a3,a4) area;
dcl a(4) area;

a(1) = a1;
a(2) = a2;
a(3) = a3;
a(4) = a4;
```

ADD, DIVIDE, and MULTIPLY do not return scaled FIXED BIN

Under the RULES(IBM) compile-time option, which is the default, variables can be declared as FIXED BIN with a nonzero scale factor. Infix, prefix, and comparison operations are performed on scaled FIXED BIN using the same semantics as the old compilers.

However, when the ADD, DIVIDE, or MULTIPLY built-in functions have arguments with nonzero factors or specify a result with a nonzero scale factor, the Enterprise PL/I compiler evaluates the built-in function as FIXED DEC rather than as FIXED BIN as the older compilers did.

For example, Enterprise PL/I compiler would evaluate the DIVIDE built-in function in the assignment statement below as a FIXED DEC expression:

```
dcl (i,j) fixed bin(15);
dcl x     fixed bin(15,2);
       ⋮
x = divide(i,j,15,2);
```

Differences in Descriptor Format

The default descriptor format in Enterprise PL/I is different from previous versions of PL/I. The CMPAT(V2) and CMPAT(V1) compiler options are available to cause Enterprise PL/I to use the previous descriptor format. This is particularly useful, for example, in applications where DB2 passes strings to Enterprise PL/I, or where Assembler programs pass descriptors in the previous format, or where Assembler programs read PL/I descriptors and expect the old format.

Differences in AMODE(24) Support

AMODE(31) and RMODE(ANY) are the default settings for the Enterprise PL/I application. Amode(24) applications are only supported for compatibility reasons. To use the AMODE(24) feature, the following is required:

1. The application program has to be compiled with the PL/I compiler option NORENT and run with the Language Environment option ALL31(OFF).
2. The application program needs to be linked with the SIBMAM24 dataset concatenated in front of the SCEELKED dataset during the link step. The support is only available through ++APAR PQ52718.

Chapter 6. Tuning your Enterprise PL/I program

After you migrate to Language Environment, you should retune your applications to maximize the performance. When you retune an application, it is not always possible to maximize CPU and storage at the same time. Often you will find that, in order to obtain better CPU, you need to use more storage, or vice versa. This chapter provides general tips to help you to retune your applications under Language Environment.

For more information on tools you can use to improve performance for your applications, see *OS/390 Language Environment Customization* or *OS/390 V2R10 Language Environment Customization*, and *Enterprise PL/I for z/OS and OS/390 Programming Guide*.

Improving CPU utilization

The following discussion shows ways to help you obtain better CPU utilization:

- Reduce the number of GETMAINs and FREEMAINs issued by Language Environment.

  Use the Language Environment RPTSTG(ON) option to produce the storage report. Specify the reported storage amount in the corresponding Language Environment storage run-time options.

- Reduce the number of LOADs and DELETEs issued by Language Environment.

  Put the commonly used Language Environment library routines in (E)LPA. The following lists the recommended candidates for Enterprise PL/I:

  – CEEBINIT (LPA)
  – CEEPLPKA (ELPA)
  – CEEEV011 (ELPA)

  See *OS/390 Language Environment Customization* for a complete list of library routines that can be put in (E)LPA.

- Avoid AMODE switching between library routines.

  Use AMODE(31) for your application, if possible, so you can specify Language Environment ALL31(ON) option. If ALL31(ON) is in effect, there is no AMODE switching among library routines.

- Excessive raising of conditions will degrade performance.

- Use DFSMS®-provided system-determined BLKSIZE.

  On OS/390, use BLKSIZE(0) for an output file that can be blocked. DFSMS determines the optimal block size for you which can improve the file performance.

- Use Language Environment Library Routine Retention facility (LRR).

  You can get a better CPU performance if you use LRR. When LRR is used, Language Environment keeps certain Language Environment resources in storage when an application ends. Subsequent invocations of programs that use LRR is much faster because the Language Environment resources left in storage are reused.

For example, you can use LRR for your IMS/DC environment to improve performance.

Note that because LRR leaves LE resources in the storage for a long period of time, you must assess your storage availability to accommodate the situation.

Improving storage utilization

The following discussion helps you to obtain better storage utilization:

- Use Language Environment option HEAP(,,ANY) option, if possible.

  For Enterprise PL/I, Language Environment will allocate the heap storage above the 16M line if the following is true:

  – The requestor is in AMODE(31)
  – HEAP(,,ANY) is in effect
  – The main program is in AMODE(31)

- Use Language Environment STACK(,,ANY) option, if possible.

  Your application must be in AMODE(31). Language Environment allocates the stack storage above the 16M line when your application is recompiled with Enterprise PL/I and linked with Language Environment.

- Reduce the IBM-supplied default values in Language Environment storage options.

  If you use a smaller value, Language Environment will allocate less storage each time, but it could result in more GETMAINs and FREEMAINs being issued.

- Put commonly used Language Environment library modules in (E)LPA.

  The library routines in (E)LPA do not occupy storage in your application region, so your application has more storage to use. See the recommended library routines for (E)LPA in “Improving CPU utilization” on page 28.

Improving performance under IMS

The following discussion helps you to obtain better performance under IMS:

Use Language Environment Library Routine Retention (LRR) facility to reduce the number of LOADs/DELETEs and GETMAINs/FREEMAINs issued by Language Environment for each transaction.

Preload commonly used Language Environment library modules and frequently used top-level applications.

Chapter 7. Subsystem considerations

This chapter discusses subsystem-specific considerations that you need to know when you migrate your applications running under CICS, IMS, and DB2.

CICS considerations

The CICS Storage Protect facility was introduced under CICS 3.3. This provides more data integrity and security for the application program and especially for the entire CICS region. Because of the new feature, you might discover that some of the successfully running OS PL/I applications start to fail with ASRA(0C4) abend and the CICS message DFHSR0622.

If the above problem occurs in your Enterprise PL/I application program, set the CICS system initialization parameter RENTPGM=NOPROTECT. This sets the protection of the user program in user key. The default for RENTPGM is PROTECT.

If PUT statements are used in your Enterprise PL/I CICS application, especially the `PUT DATA` statement, it might trigger the above error.

Remember also that in CICS programs these PUT statements are intended for debugging purposes only. They have a negative impact on performance, and we do recommend that you don't use them in production programs.

If you mix old and new object code under CICS, you must adhere to all the rules and restrictions described earlier in this book.

Updating CICS System Definition (CSD) file

When you bring up a CICS region with Language Environment, you must ensure the module names listed in Language Environment CEECCSD are defined in the CSD. You can locate CEECCSD in SCEESAMP. If you use CICS Version 4 autoinstall facility, you do not need to define Language Environment modules manually in the CSD.

In order to run a Enterprise PL/I CICS application, you need to define the Enterprise PL/I member event handler CEEEV011 in the CICS CSD definition table:

```
DEFINE PROGRAM(CEEEV011) GROUP(CEE) LANGUAGE(ASSEMBLER)
```

Macro-level interface

The CICS macro-level interface is not supported.

SYSTEM(CICS) compile-time option

The SYSTEM(CICS) option must be used when you compile your CICS programs.

Linking Enterprise PL/I applications under CICS

You are generally no longer required to take special actions when you link a Enterprise PL/I object module under CICS. However, if you did **not** specify OPTIONS(FETCHABLE) on a PROCEDURE statement for a routine that is to be FETCHed, you must code the linkage editor ENTRY statement so that it nominates the actual entry point.

Also note that prior to CICS 1.3, PDSEs are not supported. From CICS Transaction Server 1.3 onwards, there is support in CICS for PDSEs. Please refer to the CICS Transaction Server for OS/390 Release Guide, GC34-5701, where there are several references to PDSEs, and a list of prerequisite APAR fixes.

FETCHing a PL/I MAIN procedure

CICS does not support PL/I FETCHing in a PL/I MAIN procedure.

Run-time output

Run-time output is now transmitted to the CICS transient data queue CESE. Language Environment ignores the MSGFILE option under CICS. Figure 1 shows format of the output data queue.

ASA	Terminal id	Transaction id	B	DateTime YYYYMMDDHHMMSS	B	Data

Figure 1. CESE output data queue

In addition, the PL/I transient queues CPLI and CPLD are no longer used. As a result, you do not need to specify entries for the CPLI and CPLD in the CICS Destination Control Table (DCT).

Abend codes used by PL/I under CICS

The APL*x* abend codes that were issued under OS PL/I Version 2 are no longer issued. Instead, Language Environment-defined abend codes are issued. For more information about Language Environment abend codes, see *OS/390 Language Environment Debugging Guide and Run-Time Messages*.

IMS considerations

Interfaces to IMS

Language Environment supports the PLITDLI, ASMTDLI, and EXEC DLI interfaces from a PL/I routine. It also supports CEETDLI interface from a Enterprise PL/I routine running under IMS/ESA® Version 4.

Under Language Environment, CEETDLI is the recommended interface. CEETDLI supports calls that use an Application Interface Block (AIB) or a Program Communication Block (PCB). CEETDLI is available under IMS/ESA Version 4. For more information about AIB and a complete description of the CEETDLI interface, see *IMS/ESA Version 4 Application Programming Guide*.

SYSTEM(IMS) compile-time option

The SYSTEM(IMS) option should be used when compiling all PL/I MAIN programs invoked from IMS.

When you recompile your main procedure with Enterprise PL/I, the object module assumes that the parameters are passed as BYVALUE. Language Environment converts the parameters to the BYVALUE style for you, if necessary, so the parameters are always passed correctly.

If the BYADDR attribute is specified or implied for the paremeters to an IMS MAIN routine, when you compile your main procedure with Enterprise PL/I, you will receive an error message and the compiler will apply the BYVALUE attribute instead.

PLICALLA Support in IMS

The OS PL/I PLICALLA entry point is supported under Language Environment; however, it is **not** a recommended interface for IMS. Instead, use the SYSTEM(IMS) compile-time option and the CEESTART entry point.

Language Environment provides the same support for OS PL/I PLICALLA applications; however, when you recompile your main load module with Enterprise PL/I and want to continue to use PLICALLA, you must follow additional rules. See "PLICALLA considerations" on page 15 for details.

PSB language options supported

Language Environment supports PL/I applications with the following PSBGEN LANG options in the supported releases of IMS:

IMS/ESA Version 4

Table 10 shows support for PSB LANG options in IMS/ESA Version 4.

Table 10. PSB LANG options for IMS/ESA Version 4 Release 1

SYSTEM option	Entry point	LANG=
IMS	CEESTART	PLI or other values except PASCAL
IMS	PLICALLA	PLI
Omitted	CEESTART	Illegal
Omitted	PLICALLA	PLI

IMS/ESA Version 3 Re1ease 1

Table 11 shows support for PSB LANG options in IMS/ESA Version 3 Release 1.

Table 11. PSB LANG options for IMS/ESA Version 3 Release 1

SYSTEM option	Entry point	LANG=
IMS	CEESTART	PLI
IMS	PLICALLA	PLI
Omitted	CEESTART	Illegal
Omitted	PLICALLA	PLI

Assembler driving a PL/I transaction

If an assembler program is driving a transaction program written in PL/I, and assuming the PSBGEN LANG= option remains unchanged, you must use the SYSTEM(MVS) compile-time option when you recompile the PL/I main program with Enterprise PL/I. In this case, no changes to the assembler program are required.

Storage usage considerations

With IMS/ESA Version 3 Release 1, the parameters passed to the IMS interfaces are no longer restricted to the area below the 16M line. The parameters will be above the 16M line if you observe the following rules:

- If the parameters passed to IMS are in CONTROLLED or BASED storage, specify the ANYWHERE suboption of the HEAP run-time option.
- If the parameters passed to IMS are in AUTOMATIC storage, specify the ANYWHERE suboption of the STACK run-time option.
- If the parameters passed to IMS are in STATIC storage, link the load module with the AMODE(31) attribute.

Coordinated condition handling under IMS

Language Environment and IMS condition handling are coordinated, which means that if a program interrupt or abend occurs when you application is running in an IMS environment, the Language Environment condition manager is informed whether the problem occurred in your application or in IMS. If the problem occurs in IMS, Language Environment, as well as any invoked HLL-specific condition handler, percolates the condition back to IMS.

With Language Environment run-time option TRAP(ON), Language Environment continues to support coordinated condition handling for the PLITDLI and ASMTDLI interface invoked from a PL/I routine.

With IMS/ESA Version 3 with PTF UN4928 or IMS/ESA Version 4, Language Environment also supports the coordinated condition handling for CEETDLI, CTDLI from a C routine, CBLTDLI from a COBOL program, AIBTDLI from a PL/I program, and ASMTDLI from a non-PL/I program.

Note that if a program interrupt or abend occurs in your application outside of IMS, or if a software condition of severity 2 or greater is raised outside of IMS, the Language Environment condition manager takes normal condition handling actions described in OS/390 Language Environment Programming Guide. In this case, in order to give IMS a chance to do database rollback, you must do one of the following:

- Resolve the error completely so that you application can continue.
- Issue a rollback call to IMS, and then terminate the application.
- Make sure that the application terminates abnormally by using the ABTERMENC(ABEND) run-time option to transform all abnormal terminations into system abends in order to cause IMS rollbacks.
- Make sure that the application terminates abnormally by providing a modified assembler user exit (CEEBXITA) that transforms all abnormal terminations into system abends in order to cause IMS rollbacks.

The assembler user exit you provide should check the return code and reason code or the CEEAUE_ABTERM bit, and requests an abend by setting the CEEAUE_ABND flag to ON, if appropriate. See OS/390 Language Environment Programming Guide for details.

Performance enhancement with Library Retention(LRR)

If you use LRR, you can get an improvement in performance. See “Improving CPU utilization” on page 28 for details.

DB2 considerations

If you write a user-defined function in PL/I, DB/2 passes some string-locator descriptors to the PL/I procedure. These descriptors have a different format than the format used by default under Enterprise PL/I.

In order for such a program to run correctly under Enterprise PL/I, you must compile the program with the CMPAT(V2) or CMPAT(V1) option.

Appendix A. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION AS IS˜ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors.

Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those

Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX
CICS
CICS/ESA
DB2
DFSMS
DFSORT
IBM
IMS
IMS/ESA
Language Environment
MVS
OpenEdition
OS/390
RACF
System/390
VisualAge
z/OS

Intel is a registered trademark of Intel Corporation in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States and other countries.

Pentium is a registered trademark of Intel Corporation in the United States and other countries.

Unicode is a trademark of the Unicode Consortium.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product or service names may be the trademarks or service marks of others.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Bibliography

Enterprise PL/I publications

Programming Guide, SC27-1457
Language Reference, SC27-1460
Messages and Codes, SC27-1461
Diagnosis Guide, GC27-1459
Compiler and Run-Time Migration Guide, GC27-1458

PL/I for MVS & VM

Installation and Customization under MVS, SC26-3119
Language Reference, SC26-3114
Compile-Time Messages and Codes, SC26-3229
Diagnosis Guide, SC26-3149
Migration Guide, SC26-3118
Programming Guide, SC26-3113
Reference Summary, SX26-3821

z/OS Language Environment

Concepts Guide, SA22-7567

Debugging Guide, GA22-7560

Run-Time Messages, SA22-7566

Customization, SA22-7564

Programming Guide, SA22-7561

Programming Reference, SA22-7562

Run-Time Migration Guide, GA22-7565

Writing Interlanguage Communication Applications, SA22-7563

CICS Transaction Server

Application Programming Guide, SC33-1687

Application Programming Reference, SC33-1688

Customization Guide, SC33-1683

External Interfaces Guide, SC33-1944

DB2 UDB for OS/390 and z/OS

Administration Guide, SC26-9931

An Introduction to DB2 for OS/390, SC26-9937

Application Programming and SQL Guide, SC26-9933

Command Reference, SC26-9934

Messages and Codes, GC26-9940

SQL Reference, SC26-9944

DFSORT™

Application Programming Guide, SC33-4035

Installation and Customization, SC33-4034

IMS/ESA®

Application Programming: Database Manager, SC26-8015

Application Programming: Database Manager Summary, SC26-8037

Application Programming: Design Guide, SC26-8016

Application Programming: Transaction Manager, SC26-8017

Application Programming: Transaction Manager Summary, SC26-8038

Application Programming: EXEC DL/I Commands for CICS and IMS™, SC26-8018

Application Programming: EXEC DL/I Commands for CICS and IMS Summary, SC26-8036

z/OS MVS

JCL Reference, SA22-7597

JCL User's Guide, SA22-7598

System Commands, SA22-7627

z/OS UNIX System Services

UNIX System Services Command Reference, SA22-7802

UNIX System Services Programming: Assembler Callable Services Reference, SA22-7803

UNIX System Services User's Guide, SA22-7801

z/OS TSO/E

Command Reference, SA22-7782

User's Guide, SA22-7794

z/Architecture

Principles of Operation, SA22-7832

Unicode® and character representation

OS/390 Support for Unicode: Using Conversion Services, SC33-7050

Index

S

T

U

We'd Like to Hear from You

Enterprise PL/I for z/OS and OS/390
Compiler and Run-Time Migration Guide
Version 3 Release 1

Publication No. GC27-1458-00

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:

  Internet: COMMENTS@VNET.IBM.COM

  Be sure to include the following with your comments:

  – Title and publication number of this book
  – Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

Enterprise PL/I for z/OS and OS/390
Compiler and Run-Time Migration Guide
Version 3 Release 1

Publication No. GC27-1458-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

May we contact you to discuss your comments? Yes No

Would you like to receive our response by E-Mail?

_____

Your E-mail address

Name

Address

Company or Organization

Phone No.

Readers' Comments
GC27-1458-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department HHX/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5655-H31

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

Enterprise PL/I for z/OS and OS/390 Library

SC27-1456	Licensed Program Specifications
SC27-1457	Programming Guide
GC27-1458	Compiler and Run-Time Migration Guide
GC27-1459	Diagnosis Guide
SC27-1460	Language Reference
SC27-1461	Compile-Time Messages and Codes

GC27-1458-00

Spine information:

Enterprise PL/I for z/OS and OS/390 Compiler and Run-Time Migration Guide *Version 3 Release 1*